

# Uniform van Lambalgen's theorem fails for computable randomness

Bruno Bauwens\*

October 5, 2015

## Abstract

We show that there exists a bitsequence that is not computably random for which its odd bits are computably random and its even bits are computably random relative to the odd bits. This implies that the uniform variant of van Lambalgen's theorem fails for computable randomness. (The other direction of van Lambalgen's theorem is known to hold in this case, and known to fail for the non-uniform variant.)

A *martingale*  $f$  is a function mapping bitstrings to non-negative reals such that  $f(x) = (f(x0) + f(x1))/2$  for all  $x$ . A sequence  $\alpha$  is *computably random* or *CR* (respectively CR relative to  $\beta$ ) if for every computable martingale  $f$  (respectively computable relative to  $\beta$ ), the set of values of  $f$  on all prefixes of  $\alpha$ , is bounded.

Observe that the odd bits of a CR sequence define a sequence that is also CR. Suppose that the odd bits of some sequence are CR and its even bits are CR relative to its odd bits. Can we conclude that the sequence is CR? This has been an open question for a while [1, 4], and has been answered in the positive with erroneous arguments several times. Our main result answers the question in the negative. This is remarkable, because the answer is positive for the closely related notion of Schnorr randomness [3].

**Theorem 1.** *There exists a non-CR sequence for which its odd bits are CR and its even bits are CR relative to its odd bits.*

Let  $\alpha = \alpha_1\alpha_2\dots$  and  $\beta = \beta_1\beta_2\dots$  be sequences. There is a closely related positive folklore result: if both  $\alpha$  is CR relative to  $\beta$  and  $\beta$  is CR relative  $\alpha$ , then  $\alpha_1\beta_1\alpha_2\beta_2\dots$  is CR.<sup>1</sup>

Van Lambalgen's requirement for randomness is that  $\alpha_1\beta_1\alpha_2\beta_2\dots$  is random if and only if,  $\alpha$  is random and  $\beta$  is random relative to  $\alpha$ . Both for computable and Schnorr randomness this fails: there exists a random  $\alpha_1\beta_1\alpha_2\beta_2\dots$  for which  $\alpha$  is not random relative to  $\beta$  [5]. However, for both notions of randomness, this direction holds when the uniform<sup>2</sup> variant of relative randomness is used [3]. Hence, the uniform version of van Lambalgen's requirement holds for Schnorr randomness, but not for computable randomness.

---

\*National Research University Higher School of Economics, Faculty of Computer Science, Kochnovskiy Proezd 3, Moscow, 125319 Russia,

<sup>1</sup> Indeed, any martingale is the product of a martingale that only bets on odd bits and one that bets on even bits. This decomposition can happen in a computable way. The proof finishes by a simple transformation of these martingales to conditional martingales.

<sup>2</sup> A function  $f$  is *uniformly*  $\alpha$ -computable if there exists an oracle Turing machine  $U$  such that  $U^\alpha = f$  and  $U^\beta$  is total for all  $\beta$ . A sequence is uniformly random relative to  $\alpha$  if no uniformly  $\alpha$ -computable martingale has unbounded values on it.

# 1 Proof

We use computable randomness of pairs of strings. A *bivariate* martingale is a function mapping pairs of strings to non-negative reals such that for all strings  $x$  and  $y$ , the functions  $f(x, \cdot)$  and  $f(\cdot, y)$  are martingales. A *pair*  $(\alpha, \beta)$  is CR if each such computable function has bounded values on all initial segments of  $\alpha$  and  $\beta$ .

**Lemma 2** (Folklore).  *$(\alpha, \beta)$  is CR if and only if  $\alpha_1\beta_1\alpha_2\beta_2\dots$  is CR.*

*Proof.* If  $f$  is a univariate martingale, we can transform it to a two dimensional martingale  $g$ : for strings  $x$  and  $y$  of length  $n$ , let  $g(x, y) = f(x_1y_1\dots x_ny_n)$ , and for  $|x| < |y|$ , (respectively  $|y| < |x|$ ), we obtain  $g(x, y)$  by averaging over all extensions of  $x$  of length  $|y|$  (respectively of  $y$  of length  $|x|$ ).

For the other direction, transform a bivariate martingale  $g$  to a univariate one given by

$$f(x) = \begin{cases} g(x_1x_3\dots x_{|x|-1}, x_2x_4\dots x_{|x|}) & \text{if } |x| \text{ is even} \\ g(x_1x_3\dots x_{|x|}, x_2x_4\dots x_{|x|-1}) & \text{otherwise} \end{cases}$$

By the savings technique it suffices to consider martingales  $g$  such that  $g(xv, yw) \geq g(x, y)/2$  for all extensions  $(xv, yw)$  of  $(x, y)$ . By this assumption, if  $g$  is unbounded on initial segments of  $(\alpha, \beta)$  of any length, it is unbounded on initial segments of equal length, and hence  $f$  is unbounded.  $\square$

By this lemma, Theorem 1 is equivalent to:

**Proposition 3.** *There exists a pair  $(\alpha, \beta)$  of sequences that is not CR such that  $\alpha$  is CR and  $\beta$  is CR relative to  $\alpha$ .*

We use a technique from the proof of the Gács–Kucera theorem (see [2, Lemma 8.3.1, p325]) to define a random sequence that diagonalizes against a sequence of tests and at the same time encodes some information. For this technique, we use a technical lemma.

**Lemma 4.** *For each martingale  $d$ , each string  $x$  and each natural number  $i$ , there exist at least two strings  $y$  of length  $i + 2$  such that  $d(xy)/d(x) < 1 + 2^{-i}$ .*

*Proof.* For all  $0 < \varepsilon < 1/2$ , at most a fraction  $1/(1 + \varepsilon)$  of strings  $y$  of a given length have  $d(xy)/d(x) \geq 1 + \varepsilon$ . Hence, a fraction  $1 - 1/(1 + \varepsilon) > \varepsilon/2$  does not have this property. The amount of such strings of length  $i + 2$  is at most  $2^{i+2}\varepsilon/2$  which equals 2 for  $\varepsilon = 2^{-i}$ .  $\square$

*Proof of Proposition 3.* Note that it suffices to only consider rational martingales, because they define the same class of CR-sequences [2, Prop. 7.1.2 p270]. We choose  $\alpha$  to be any computably random sequence. Let  $d_1, d_2, \dots$  be an enumeration of all rational partial martingales that are partial computable with oracle  $\alpha$ . Let  $\varepsilon_s = 2^{-s}$ .  $\beta$  is constructed in stages, together with a total martingale  $d$ .

Initially,  $\beta$  is the empty string and  $d = 1$ . At each stage  $s$  we update  $\beta$  and  $d$ :

- If  $d_s$  is total and  $d(\beta)$  is positive, we replace  $d$  by  $d + \frac{\varepsilon_s}{d_s(\beta)}d_s$ . Otherwise,  $d$  is unchanged.

- To  $\beta$  we append either the lexicographically first or second string  $y$  of length  $2^{s+2}$  such that

$$\frac{d(\beta y)}{d(\beta)} < 1 + \varepsilon_s \quad (1)$$

(by Lemma 4 there exist at least two such strings). The choice is determined by the totality of  $d_{s+1}$ .

- To  $\beta$  we again append either the lexicographically first or second string  $y$  of length  $2^{s+2}$  such that the equation above holds. The choice now corresponds to the value of  $\alpha_t$  for  $t$  being the maximal of all computation times and all uses of the oracle  $\alpha$ , in the evaluations of  $d(z)$  on strings  $z$  of length at most  $|\beta y|$ .

We assume that in each stage the value of  $t$  is strictly above the value of  $t$  in the previous stage. (This is natural, because after each stage,  $d$  becomes more complex and needs to be evaluated on a larger set of strings.) End of construction.

We show that  $\beta$  is computably random relative to  $\alpha$ . Observe that the value  $d(\beta)$  after each stage  $s$  is at most  $(1 + \varepsilon_1 + \dots + \varepsilon_s) \exp(2\varepsilon_1 + \dots + 2\varepsilon_s)$ , and this has the finite limit  $2 \exp 2$ . Indeed, after a possible update of  $d$ , the value  $d(\beta)$  increases by  $\varepsilon_s$ , and after each extension of  $\beta$ , the value increases by at most a factor  $1 + \varepsilon_s \leq \exp \varepsilon_s$ .

For each test  $d_s$  that is computable relative to  $\alpha$ , we have that  $d_s \leq O_s(d)$ , because at stage  $s$ , this test will be used to increment  $d$ . Hence,  $\beta$  is computably random relative to  $\alpha$ .

It remains to show why the pair is not computably random. We construct a bivariate martingale  $e$ . The martingale  $e$  plays only on the bits  $\alpha_t$  that were used in the construction and on these bits the capital is doubled. To compute these bits (and the values  $t$ ), we need to know the function  $d$  used in each stage, and need to evaluate it on all strings of some lengths. For this, we need to know  $\alpha$  and which functions  $d_1, d_2, \dots$  are total. The key observation is, that knowing  $\beta$  we can decide totality of the functions  $d_1, d_2, \dots$ . Indeed, knowing the (updated) function  $d$  in a stage  $s$ , we can compute the lexicographically first and second string for which (1) holds, then observe which one equals the corresponding segment of  $\beta$ , and from this we know the totality of  $d_{s+1}$ . This allows us to update  $d$  in the next stage, and we can repeat this procedure. In the evaluation of  $e(a, b)$  we use  $a$  and  $b$  as initial segments of  $\alpha$  and  $\beta$ .

*Detailed construction of  $e$ .* Let  $e(\text{empty string}, b) = 1$  for all  $b$ . To evaluate  $e(a, b)$ , we assume that  $b$  is long enough to go through the procedure below.<sup>3</sup> If this is not the case, we compute the value by averaging over all extensions of  $b$  of some long enough length. We start with  $s = 0$  and  $d = 1$ .

At stage  $s$  we evaluate the current function  $d$  on all strings of length at most  $2 \sum_{i=0}^s (i+2)$ . Let  $t$  be the maximal computation time for such a string. Note that  $t$  can be infinite if one of the guesses for  $d_1, \dots, d_s$  was wrong. We only need to discriminate between the following cases:

- If  $t < |a|$ , we check whether the appropriate segment of length  $s+2$  of  $b$  is indeed the lexicographically first or second string  $y$  satisfying (1). If this is not case, we set  $e(a, b) = 1$  (the value of  $e$  does not matter here); otherwise, we proceed to stage  $s+1$  with the function  $d$  updated corresponding to the suggested totality of  $d_{s+1}$ .

---

<sup>3</sup> It suffices that  $|b| \geq 2 \sum_{i=0}^{|a|} (i+2)$ . Indeed, the number of stages that need to be evaluated is at most  $|a|$  because the sequence of values of  $t$  is strictly increasing.

- If  $t > |a|$  (or  $t$  is infinite), we set  $e(a, b) = e(a_1 \dots a_{|a|-1}, b)$ .
- Otherwise (i.e. if  $t = |a|$ ), let  $i$  be the bit that is encoded in the subsequent part of  $b$  of length  $s + 2$  (which is supposed to encode  $\alpha_t$ ). Now set:

$$e(a, b) = \begin{cases} 2e(a_1 \dots a_{|a|-1}, b) & \text{if } a_{|a|} = i \\ 0 & \text{otherwise.} \end{cases}$$

$e$  is computable, and on initial segments of  $\alpha$  and  $\beta$ , the martingale  $e$  is unbounded.  $\square$

## Acknowledgments

I am grateful to Jason Rute for useful discussion and for bringing this question under attention. I am grateful to the Heidelberg university for financial support through the “Focus Semester on Algorithmic Randomness” in June 2015. I thank the organizers, Wolfgang Merkle, Nadine Losert, Martin Monath and the participants for creating a nice work atmosphere.

## References

- [1] David Diamondstone, Noam Greenberg, and Dan Turetsky. A van Lambalgen theorem for Demuth randomness. In *Proceedings of the 12th Asian Logic Conference*, pages 115–124, 2013.
- [2] R.G. Downey and D.R. Hirschfeldt. *Algorithmic Randomness and Complexity*. Theory and Applications of Computability. Springer, 2010.
- [3] Kenshi Miyabe and Jason Rute. Van Lambalgens theorem for uniformly relative schnorr and computable randomness. In *Proceedings of the 12th Asian Logic Conference*, pages 251–270. World Scientific, 2013.
- [4] Andre Nies. Logic blog 2014. *Preprint arXiv:1504.08163*, 2015.
- [5] Liang Yu. When van Lambalgen’s theorem fails. *Proceedings of the American Mathematical Society*, pages 861–864, 2007.